HELSINKI UNIVERSITY OF TECHNOLOGY
Automation Technology Laboratory

ILONANI

A table-top robot for remote interaction

Department of Automation and Systems Technology
Helsinki University of Technology
Gonzalo Zubieta García
2009

Información de interés

Autor: Gonzalo Zubieta García

Tutor: Panu Harmo

Lugar de realización: Universidad Tecnológica de Helsinki, Finlandia.
Departamento de sistemas automáticos.

Coordinador académico: Francisco José Rodríguez Urbano

Fecha de lectura: 16 de Junio de 2009

Calificación obtenida: Notable (3/5 calificación en HUT)

Descripción general

RESUMEN

Este documento presenta los resultados de la investigación realizada sobre el desarrollo de un robot de sobremesa de interacción remota. Este robot de sobremesa fue concebido para incluir una tableta de internet **Nokia N810** a modo de "cabeza y cara" del robot, concentrando las funciones del procesador principal y la interfaz con los usuarios. Este **N810** se sentaría en una base que le proporcionaría el movimiento controlado de giro e inclinación a través de unos microcontroladores y servomecanismos.

La labor realizada bajo este proyecto se centra en el desarrollo de dos soluciones que ayudan a completar el desarrollo del robot.

El **primer objetivo** de la versión actual fue desarrollar una placa de control, incluyendo un microcontrolador y los programas informáticos necesarios para controlar un par de servomotores, que proporcionan al robot un mecanismo de giro e inclinación, lo que permitiría el movimiento del **N810**.

El **segundo objetivo** era crear una comunicación entre la **N810** y la placa de control para conseguir controlar los movimientos de los servos a través de la tableta de Nokia, lo que permitiría la creación de una interfaz muy amigable con el mundo exterior.

El documento comienza con una descripción del contexto en el que este robot se ha concebido: Un mundo interconectado con dispositivos inteligentes y de ambiente, donde las personas esperan ser capaces de telecomunicarse en cualquier lugar, en cualquier momento.

La labor realizada bajo este proyecto se ha centrado en el desarrollo de los diferentes códigos de programación para conseguir cada uno de los dos objetivos, y su rendimiento ha sido comprobado haciendo varias pruebas para obtener una primera versión del software libre de problemas y errores.

Los resultados del proyecto son dos:

- Puesta en marcha de un sistema para controlar dos servomotores a través de una placa de control y creación de un mecanismo de giro e inclinación.
- Creación de una comunicación entre el **N810** y la placa de control para gestionar el movimiento del robot con facilidad desde el **Nokia**.

INTRODUCCIÓN

No es nuevo para nosotros el hecho de que la población en los países desarrollados está envejeciendo. Se espera que esta tendencia también aparezca en los países en vías de desarrollo. Los ancianos representarán un porcentaje mayor respecto a toda la sociedad, y la sociedad debe responder desarrollando servicios que presenten accesos más fáciles y cómodos para las personas mayores. Esto se consigue creando dispositivos amigables con interfaces simples e intuitivas.

Éste es el punto de partida para el proyecto **ILONANI** ("Mi Bienestar"), que busca desarrollar un robot de sobremesa de interacción remota, además de habilitar una comunicación sencilla entre el robot y el usuario. **ILONANI** también podría facilitar a los cuidadores la monitorización y el conocimiento de las necesidades de los ancianos.

El programa de reconocimiento de voz puede significar una gran ayuda tanto para las personas discapacitadas por alguna enfermedad como para las personas discapacitadas por el paso de los años. Las órdenes dadas mediante la voz pueden ser convertidas en comandos para controlar diferentes dispositivos. Por ejemplo, con un simple comando de voz, los ancianos pueden hacer una llamada telefónica, encender y apagar las luces, la calefacción, etc., lo que significa dotarles de más autonomía y de la posibilidad de realizar actividades por ellos mismos, sin ningún tipo de ayuda externa.

En el futuro, la mayoría de los lugares donde pasamos más tiempo, como nuestro hogar o nuestro lugar de trabajo, estarán equipados con varios dispositivos inteligentes o sistemas de interacción entre usuarios y dispositivos, incluso entre usuarios y el edificio. Muchos de esos elementos están diseñados para conectarse entre ellos, y con una amplia red de comunicaciones como Internet. Muy pronto los dispositivos tendrán más servicios en tamaños más pequeños, como un teléfono móvil.

A través de esta interconexión entre los diferentes dispositivos nace la posibilidad de una comunicación global, donde el intercambio de información alcanzará casi todo los rincones de una manera rápida y sencilla. Cada vez más personas utilizan Internet para almacenar información, ya sea personal o no, a través de redes sociales, blogs o páginas web. Se descubrirán con absoluta certeza nuevas formas de entretenimiento, formas a las que las personas pronto se acostumbrarán, pero que hoy en día ni se imaginan que puedan existir. Esto ya ha ocurrido numerosas veces, incluyendo la creación del sistema de gestión de videos llamado **Youtube**, el sistema de almacenamiento de fotos **Picasa**, o los servidores para almacenar datos de cualquier tipo (películas, música, libros…) como **Rapidshare**.

Volviendo al proyecto **ILONANI**, el dispositivo se basa en una aplicación de videoconferencia guiada por voz para hacer posible la interacción entre el dispositivo y los ancianos. Hoy en día, el uso de cámaras guiadas por voz es una realidad. El uso principal dado a estos sistemas forma parte de las comunicaciones orientadas a los negocios. Estos sistemas son utilizados en

reuniones o videoconferencias donde varias personas están involucradas a cada lado de la llamada, y puede funcionar de modo punto a punto (llamada entre dos lugares) o multipunto (llamada entre más de dos lugares).

Estos sistemas consisten en dos partes fundamentalmente:

- Detección de la fuente de sonido.
- Orientación de la cámara para enfocar dicha fuente.

Para lograr la detección de la fuente de sonido es necesario colocar estratégicamente una serie de micrófonos y desarrollar un complejo algoritmo que pueda filtrar y eliminar la influencia de las reverberaciones, ecos y ruidos, para así recibir la fuente de sonido de manera clara y obtener el vector de dirección de dicha fuente.

A la hora de orientar la cámara en esa dirección se utilizan varios servomotores que reciben las señales de control de un microcontrolador y dirigen la cámara en la dirección de la fuente.

Las ventajas de un **sistema de videoconferencia** no sólo son aplicables al campo de los negocios, también puede reportar beneficios en ámbitos como las universidades, bancos, instituciones financieras, compañías de seguros, institutos de formación, las multinacionales, pequeñas y grandes empresas y organismos públicos.

También es posible aplicar la videoconferencia para hacer más productiva y eficiente la relación entre los cuidadores y los ancianos. Es posible saber en qué habitación de la casa se encuentra la persona, y si ha sufrido una caída o cuáles son sus signos vitales. Todo ello sin necesidad de que los ancianos envíen una petición de ayuda directamente; **el dispositivo puede hacerlo por ellos.**

CONTROL Y COMUNICACIÓN

Este proyecto trata de resolver dos cuestiones fundamentales claramente diferenciadas en el desarrollo general del robot:

- Control de los movimientos del robot.
- Comunicación entre **N810** y **AVR** (placa de control).

Dado que las dos secciones representan diferentes objetivos y problemas, la resolución se ha hecho por separado. Ambas soluciones incluyen código de programación.

La primera sección se refiere a la parte relativa al control de los movimientos del robot, que es la parte mecánica. Los servomotores mueven el "rostro" del robot con un mecanismo de giro e inclinación. Los servos son necesarios en el momento en que el robot debe moverse en la dirección correcta si la fuente de voz cambia su ubicación. Si el **N810** es la "**cara**" del

robot, los **motores** son los "**músculos**". Esta sección también explica por qué fue elegida la placa **AVR** en lugar de utilizar otras opciones, e incluye una breve descripción de su funcionamiento, los componentes que lo forman, el pin-out, etc.

En el segundo apartado se **desarrolla** un **software** que permite la comunicación de información y comandos entre **N810** y **AVR**. Utilizando de nuevo la comparación humano-robot, **N810** también es el "**cerebro**" del robot. El **N810** es el responsable de enviar las señales adecuadas al **AVR** para mover los servos correctamente.

CONTROL DE LOS MOVIMIENTOS DEL ROBOT

Para mover el **N810** se ha estudiado el uso de dos servomotores **FUTABA S3010**. Los servos **Futaba** son muy eficientes y tiene**n** una rápida respuesta a las necesidades tanto de velocidad como potencia en cualquier equipo. Con dos servos es posible construir un mecanismo de giro e inclinación, y cubrir una gran superficie del espacio, lo suficiente como para seguir a la fuente de sonido dentro de un rango determinado. Estos servomotores tienen tres entradas, dos de ellas para el suministro de energía (alimentación y masa) y otra para el control. La alimentación de los servos, en nuestro caso, es de **5 voltios**. Mediante la entrada de control, el **AVR** envía impulsos eléctricos para mover el servo a la posición deseada.

SELECCIÓN DE LA PLACA DE CONTROL

Para elegir la placa de control de los servos, varios compañeros de trabajo recomendaron dos placas por encima del resto: **PICSERVO**, desarrollada por una empresa de Tampere y **AVR**, desarrollado en HUT (Universidad Tecnológica de Helsinki), en el departamento de sistemas de automatización. La razón para elegir entre estas dos placas fue que las personas del departamento ya habían trabajado con ellas anteriormente, y siempre obtuvieron buenos resultados. Además, si surgiera cualquier problema, los compañeros tenían experiencia suficiente para ser capaces de ayudar a resolverlo.

PICSERVO, la placa verde, ya viene programada de fábrica y varias personas han trabajado con ella. En cuanto a la cuestión del control de servos funciona muy bien, pero tiene el inconveniente de no poder modificar el código interno del microcontrolador en caso de necesitarlo. Es simple de utilizar y se alcanzan a la perfección los objetivos solicitados.

Analizando la forma en que el dispositivo **N810** debe estar conectado con la placa de control, el **N810** funciona con **Linux** en lugar de **Windows**, y además **utiliza** el puerto de comunicación **USB**, del tipo **micro USB**. Así surge la necesidad de adaptar el sistema de **Windows/RS232** a **Linux / USB**.

La configuración **Windows/USB** se implementó mediante un convertidor **RS232/USB y no funcionó:** el valor que se lee desde la patilla **Tx** (pin de transmisión del puerto RS232) cuando se envía un pulso de la información utilizando el convertidor es significativamente menor que si no se utiliza el convertidor, por lo tanto, la placa **PICSERVO** no lo detecta como un **1 lógico**, y los servos no se mueven.

Es después de este análisis cuando **se decide utilizar el AVR**, la placa roja, que tiene muchas más posibilidades y flexibilidad **en** la programación.

PLACA DE CONTROL AVR

> "La placa **AT90USB162** ha sido especialmente diseñada para el grupo **Space Master (2008)** en **HUT**. La placa es barata (coste menor de 20 € / cada una) y se puede programar directamente desde el puerto **USB**. El chip **AT90USB162** también tiene un regulador interno de **3,3 V.** Normalmente, el dispositivo utiliza también el puerto **USB** como una fuente de alimentación.
>
> La placa consiste únicamente en los componentes esenciales del dispositivo **AT90USB162**. La tensión se puede seleccionar a **5 V** (directamente desde el puerto **USB**) o **3,3 V** desde el interior del regulador. El cristal de **16 MHz** externo es necesario para la comunicación **USB** (gestor de arranque) para que trabaje como se desea."

EL PROGRAMA DE CONTROL

Para programar la placa **AVR** se ha utilizado un código desarrollado por la empresa de microcontroladores **ATMEL**. El software ha sido utilizado antes muchas veces y su funcionamiento se ha verificado sin problemas. Los problemas iniciales se han solucionado con el tiempo y con la ayuda de la experiencia.

> "La aplicación se basa en dos tareas diferentes:
>
> - El **usb_task** (usb_task.c asociado al archivo de origen), es la tarea que realiza el proceso de enumeración de bajo nivel del puerto **USB** en modo dispositivo. Una vez que esta tarea ha detectado que la conexión **USB** está plenamente operativa, se actualiza el estado de diferentes "banderas" ("flags") que pueden ser comprobadas en las tareas de aplicaciones de alto nivel.
> - La tarea **hid** realiza la operación de la aplicación de alto nivel del dispositivo. Una vez que el dispositivo está totalmente enumerado (**DEVICE SET_CONFIGURATION solicitud recibida**), la tarea comprueba si la información ha sido recibida en su punto

final de salida, y si ha sido transmitida en su punto final de entrada."

Este **software** es **independiente** del **sistema operativo** del equipo o dispositivo que se utilice para trabajar con el **AVR**, el software es muy robusto y supone una solución ideal para el problema que aquí se trata.

Para mover el servo a la posición deseada se utiliza una señal **PWM**, y es generada por el microcontrolador del **AVR**. Variando el ancho de pulso de la señal **PWM**, es decir, variando el ciclo de trabajo, se consiguen las distintas posiciones que los servos son capaces de alcanzar dentro de un cierto rango. El **rango cubierto** por el servo es de aproximadamente **200 grados**. Este valor será suficientemente amplio si el robot se coloca cerca de una pared, es decir, si el robot no se encuentra en el centro de la habitación.

Matti Öhman, de la Universidad Tecnológica de Helsinki, explica la generación de las **señales PWM y SWP** para controlar los servos de la siguiente manera:

"Para controlar la generación de la señal **PWM** deben utilizarse los siguientes comandos:

- **$PWM,1000,1500,2000***
- **$SWP,1000,1500,2000***

El primer comando fijará el ancho del pulso de la señal **PWM a 1000us, 1500us y 2000us** para los **canales 1, 2 y 3**, respectivamente. El ancho de pulso por defecto es de 1500us en el momento del encendido. Especificaciones de **valores superiores a 20000us** dará como resultado un comportamiento **indefinido** (**PWM> 100%**) y no es recomendable.

La **diferencia** entre los comandos **PWM y SWP** es que el primero establece los valores indicados instantáneamente, mientras que el segundo utiliza rampas lentas (o barridos) para la transición a las longitudes de pulso deseadas. La **tasa de variación** para las **longitudes de pulso** es **50us/s**, por lo que tardará 20 segundos para llegar de 1000us a 2000us.

También es posible utilizar las **versiones cortas** de los comandos:

- **$PWM,1000,,* // establece de inmediato el primer canal a 1000us**

- **$SWP,,,2000* // establece gradualmente el tercer canal a 2000us**"

Los valores del ancho de pulso han sido verificados, de modo que es posible especificar valores que pertenezcan al rango recomendado (400us – 2350us).

COMUNICACIÓN ENTRE N810 Y AVR

El dispositivo **N810** es el "**cerebro**" del robot. Es el elemento responsable de la **interacción con el mundo exterior** y de transformar las órdenes dadas en comandos capaces de mover los servomotores del robot. La comunicación entre la **N810** y el **AVR** es esencial para que el robot pueda alcanzar su objetivo de una manera satisfactoria.

Gracias a la selección de la placa **AVR** en lugar de la placa **PICSERVO**, es posible trabajar directamente a través de los puertos **USB** del ordenador, sin utilizar ningún convertidor **RS232/USB**. Así, en el momento de trabajar finalmente con el dispositivo **N810**, el cambio en el código será mínimo, ya que la funcionalidad de la configuración **Linux/USB** será aplicada y verificada.

EL PROGRAMA DE COMUNICACIÓN

El código del programa para el control de los servomotores ha sido desarrollado por **Gonzalo Zubieta en HUT**, y ha sido probado con éxito en un **PC con Linux**. Las primeras pruebas para implementar el software en el **N810** están siendo llevadas a cabo al mismo tiempo que este documento está siendo escrito. Por lo tanto, los resultados hoy en día no son aún satisfactorios y el código final puede ser modificado a partir del código original para corregir los errores no detectados durante la operación a través del PC.

El código está compuesto de las siguientes funciones:

- **Apertura del puerto USB** del dispositivo para enviar y recibir señales a través de él.
- **Escritura de los comandos** para controlar los servos como exija el movimiento.
- **Creación de un bucle para continuar escribiendo** comandos, la selección de los canales y el control de los servos.
- **Cierre el puerto USB** y terminación de la comunicación.

El código está escrito y comentado con todo detalle en el apartado correspondiente del proyecto.

RESULTADOS Y SOLUCIONES ALCANZADAS

Los principales resultados de este proyecto son dos:

- **El control de los servomotores** utilizando la placa del microcontrolador **AVR**.

- **El desarrollo de una aplicación** para manejar los servomotores desde un dispositivo que funciona con Linux.

Gracias a la labor realizada en este proyecto se ha logrado desarrollar y probar una solución para el control de los servomotores, responsables del movimiento del **N810**. En la elección de la placa de control **AVR** basada en la experiencia del departamento de Automatización y Tecnología de Sistemas de **HUT**, se ha desarrollado un software para programar el microcontrolador de la placa y lograr el objetivo propuesto inicialmente. A través de algunos comandos básicos el **AVR** es capaz de generar las ondas necesarias para controlar los servos y moverlos a la posición deseada.

Por otra parte, se ha desarrollado una aplicación para hacer más fácil el control de los servos de un dispositivo que funcione con **Linux**. **El control de los servos desde un PC se ha realizado con éxito**. También era necesario crear una comunicación entre el **N810**, que trabaja con **Linux,** y el **AVR**.

Es aquí donde aparece el problema principal cuando se intenta establecer la comunicación. El **N810** es un dispositivo **periférico**, aunque con un software sencillo puede convertirse en el “anfitrión” (“host”) de la comunicación. Cuando se utiliza el **N810** para enviar las señales, y no un PC, el **AVR** debe ser alimentado externamente mediante una fuente de alimentación. El problema reside en el cable utilizado para la comunicación física. El cable es del tipo "periférico" en lugar de ser "anfitrión". Es decir, al intentar establecer una comunicación, la **energía fluye desde el AVR al N810**, y no en el sentido contrario, como debería ocurrir. Por lo que el **N810** se bloquea y se reinicia una y otra vez. La solución a este problema es comprar el cable correcto o modificar el actual para que éste pueda trabajar en el modo correcto.

TRABAJO FUTURO

La visión de **ILONANI**, un robot de sobremesa de interacción remota, puede que tenga que ser revisada. Sin embargo, si se considera actual cuando se vaya a llevar a cabo un nuevo trabajo, podría ser útil seguir el trabajo realizado en este proyecto, ya que han sido descubiertos muchos caminos equivocados y mucho del trabajo hecho puede ser reutilizado.

Además, el código y la experiencia resumida en este documento podrían ser reutilizadas en la codificación de una aplicación final, con el fin de ahorrar tiempo y esfuerzo.

COMPONENTES DEL SISTEMA

En el capítulo de descripción de los componentes del sistema se ha discutido acerca de varios bloques que forman parte del robot, pero que no han sido desarrollados en este proyecto. Por un lado se ha barajado colocar toda la alimentación y la adaptación de ella en una misma placa electrónica. El

desarrollo de esta placa necesita más investigación para diseñar y optimizar su funcionamiento, maximizando el rendimiento y reduciendo al mínimo las pérdidas de energía en el robot.

Por otra parte también se ha discutido la necesidad de utilizar una batería para alimentar el robot durante un tiempo limitado, convirtiéndose así en un elemento inalámbrico. Un análisis preliminar concluyó los valores aproximados de la capacidad y la potencia para el sistema, pero este análisis debe revisarse, ya que todavía no se ha trabajado con el sistema completo, ni con la demanda real de potencia.

Por último, se consideró la implementación de unos botones, ya sea mediante hardware o software, para controlar los movimientos del robot de forma manual. Este bloque necesita ser completamente desarrollado, construyendo los botones si fueran del tipo hardware, o desarrollando un nuevo código si se utilizan los botones software. En ambos casos se requerirá la implementación de un código para gestionar las órdenes y comunicarlas a los servomotores a través de la placa de control.

BLUETOOTH. LA COMUNICACIÓN ALTERNATIVA

Para transferir información desde y hacia el **Nokia N810**, la conexión **Bluetooth** también está disponible. Este proyecto explora la comunicación por **USB**, pero vale la pena señalar que ésta no es la única comunicación posible entre el **N810** y el **AVR**.

El proyecto **MAEMO** explica la comunicación **Bluetooth** de la siguiente manera.

> "El dispositivo **N810** tiene una radio **Bluetooth** incorporada, que permite hacer cosas como "**atar**" a un celular para realizar una conexión de datos, transferir archivos entre dispositivos y conectar dispositivos de entrada.
>
> Hay dos protocolos de red Bluetooth. El primero, **DUN** (Dial-Up Networking), está bien soportado en **MAEMO** y es el método recomendado para la "**atadura**" a un dispositivo celular. El segundo, **PAN** (Personal Area Network), no tiene apoyo oficial, pero está bien soportado por la comunidad. El **PAN** es más rápido y más versátil, pero con el apoyo de menos dispositivos que **DUN**.
>
> **DUN** es el método principal para "**atar**" a los dispositivos celulares. Es fácil de instalar y fácil de usar, pero más lento que **PAN** en muchos casos. **PAN** es más rápido y más versátil que **DUN**, aunque con el apoyo de menos dispositivos, y no con el apoyo oficial en **MAEMO**, pero con el apoyo está proporcionado por la comunidad. **PAN** es particularmente útil

para la creación de redes **Bluetooth** entre el ordenador y su tableta.”

Para utilizar esta comunicación es necesario proporcionar un sistema de comunicación **Bluetooth** al **AVR**. La versión actual del **AVR** carece de este sistema, pero se propone como un posible trabajo futuro, o una versión más evolucionada de la comunicación objeto de este proyecto, desarrollar el código que permita el enrutamiento con éxito de órdenes entre **N810** y **AVR**.

Abstract

This document presents the results of the research done on the topic of developing a table-top robot for remote interaction. This table-top robot was conceived to include a Nokia Internet Tablet N810 as its “head and face”, concentrating the functions of main processor and interface with users. This N810 would sit on a base providing controlled “turn-and-tilt” movement through some microcontroller and servomechanisms. The work done under this project concentrates on the development of two solutions that help to complete the global making up of the robot.

The first goal in the current version was to develop a microcontroller board and the software needed to control a couple of servo motors to provide a “turn-and-tilt” mechanism to the robot, which would enable the movement of the N810.

The second goal was to create a communication between the N810 and the microcontroller board to control the movements of the servos through the Nokia tablet, thereby enabling the creation of a friendly interface with the outside world.

The document starts with a description of the context in which this robot is conceived: an interconnected world of smart and ambient devices where people expect to be able to tele-communicate anywhere, any time.

The work done under this project has focused on developing of different programming codes for solving each of the two objectives, and its performance has been tested doing several tests to obtain an initial software version free of problems and errors.

The results of the project are double: on one hand, a system was put in place to control two servo motors through a microcontroller board and to create a “turn-and-tilt” mechanism; on the other, the creation of a communication between the N810 and the microcontroller board to manage the movement of the robot easily from the Nokia.

Table of Contents

1. Introduction

It’s not new to us the fact that the population in developed countries is becoming older. This trend is expected to appear in developing countries as well. Elderly people will occupy a greater percentage of society, and the society must respond by developing services that involve a more comfortable and easy access for elderly people [6]. This is achieved by creating friendly devices with simple and intuitive interfaces.

This is the starting point for the Ilonani project, which aims to develop a table-top robot for remote interaction, thus enabling an easier communication between the robot and the user. Ilonani could also facilitate to the caregivers the monitoring and the knowledge of the needs of elderly.

The voice recognition software can mean a great help both the disabled by illness or by the passage of years. The orders given by voice can be converted into commands to control different devices. For example, with a simple voice command elderly people can start a phone call, turn on and off the lights, the heating, etc., which means giving them more autonomy and the ability to perform activities by themselves, without external assistance [7].

In the future, most of the places where we spend most of our lives, as our home or our workplace, will have several smart devices or interaction’s systems between users and devices, even between users and the building [1]. And many of these elements are designed to connect among themselves, and with a wider communications network like the Internet. Soon the devices will have more and more services in smaller spaces, such as a mobile phone.

Through this interconnection between different devices comes the possibility of a global communication, where the exchange of information will reach almost every corner in a quick and easy way. More and more people use the Internet to store information, whether personal or not, through social networks, blogs and web sites. New forms of entertainment will be discovered with absolute certainty, forms that people will soon get used to work with it, but nowadays they cannot imagine it exists. This has already happened numerous times, including the creation of the video management system called YouTube, the photo storage system Picasa, or the servers for storage data of all kinds (movies, music, books...) as Rapidshare.

Turning to the Ilonani draft, the device relies on an application of voice tracking videoconferencing to make possible the interaction between the device and the elderly people. Nowadays, the use of voice tracking cameras is a reality. The main use given to these systems join the field of the connected to the communications business. These systems are used in meetings or

videoconferencing where several persons are involved at each side of the call, and it can operate point to point (call between two places) as multipoint (call between more than two places).

Illustration 1: Video conferencing [10]

These systems consist of two fundamental parts: detection of the sound source and orientation of the camera to focus on that source [5]. To achieve the detection of the sound source is necessary to place strategically a series of microphones and develop a complex algorithm that can filter and eliminate the influence of reverbs, echoes and noises, in order to receive the sound source in a clear way and obtain the vector with the direction of that source. At the time to orient the camera in that direction are used several servomotors that receive the control signals from a microcontroller and direct the camera in the direction of the source.

The advantages of a videoconferencing system are not only applicable to the field of business, it can also get benefits in areas such as universities, banks, financial institutions, insurance companies, training institutes, multinationals, small and large companies and public bodies [5]. It is also possible to apply the videoconferencing to make more productive and efficient the relationship between the caregivers and the elderly people. It is possible to know in which room of the home is located the person, and whether he had suffered a fall or what are his vital signs. All of this without the need of sends a call for help directly from the elderly; the device can do it for him.

2. Control and Communication

2.1 Introduction

This project seeks to resolve two key issues clearly differentiated within the overall development of the robot:

- Control of the robot's movements
- Communication between N810 and AVR

Since the two sections represent different objectives and problems, the resolution has been done separately. Both solutions include a specific programmed code which will be explained and commented later.

The first section refers to the part concerning the control of the movements of the robot; this is the mechanical part, the servo motors that moves the "face" of the robot with a mechanism of turn-and-tilt. The servos are necessary at the time the robot needs to move in the right direction if the voice source changes its location. If the N810 is the "face" of the robot, the motors are the "muscles". Basically, this section focuses on developing software for programming the AVR's microcontroller and thus achieving the objectives.

It also explains why the AVR board was chosen instead of using other options, and a briefly describe of its operation, the components that form it, the pin-out, etc.

In the second section it's developed a software that enables communication of information and commands between the N810 and the AVR. Using again the comparison robot-human the N810 is also the "brain" of the robot. The N810 is the responsible for sending the right signals to the AVR microcontroller to move the servos properly.

The whole descriptions of the system's components are in the appendix A. These components are the Nokia's internet tablet (N810), the AVR microcontroller board, the power supply board, the servos, the battery and the buttons.

2.2 Control of the robot's movements

One of the main objectives of the overall project is the robot's ability to orient the N810 towards a certain direction, toward a sound source, while a call is made via videoconferencing. Hence the need to design a driving block that enables the movement of the robot.

To move the N810 has been studied to use two Futaba S3010 servo motors, described in the second chapter. Futaba Servos are very efficient and has a fast response to current needs for both speed and power on any equipment. With two servos it is possible to construct a mechanism of turn-and-tilt, and to cover a large area of space, enough to keep the sound source within a certain range. These servo motors have three entries, two of them for power supply and another for control. The power of the servos, in our case, is between 5 volts and ground. Through the control input the AVR sends electrical pulses to move the servo to the desired position.

SELECTION OF THE CONTROL BOARD

To choose the board to control the servos several fellow worker recommended me two boards above the rest: PICSERVO, developed by a company in Tampere and AVR, developed in HUT, in the automation systems department. The reason for choosing between these two boards was that people in the department had already worked with them before, and always had good results. Also, if any problems arose they had enough experience to be able to help solving it.

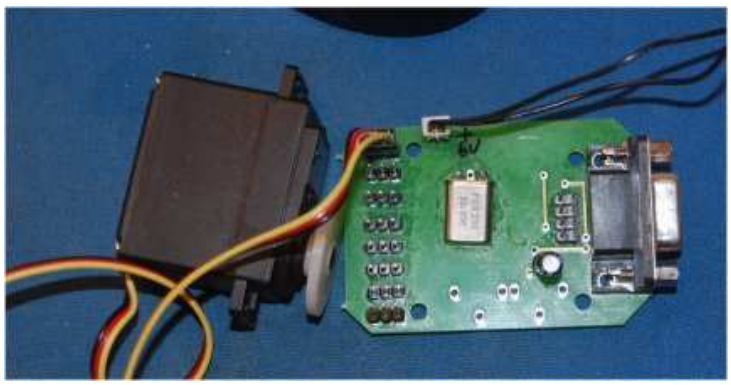


Illustration 2: PICSERVO board [18]

PICSERVO, the green board, is already programmed from the factory and several people have worked with it. On the issue of control of servos it works very well, but has the disadvantage of not being able to modify the code inside the microcontroller in case of need it. It is simple to use and achieve

perfectly with the objectives requested. Moreover, by acquiring it from the company, the board brings a very simple software for Windows, very easy to use.

The board is connected to the computer through the serial port, connection type RS232. The board requires a supply of 6 volts and a power of slightly more than 10 watts when the torque requested is high. The board supplies the servos directly, without to need external power for them.

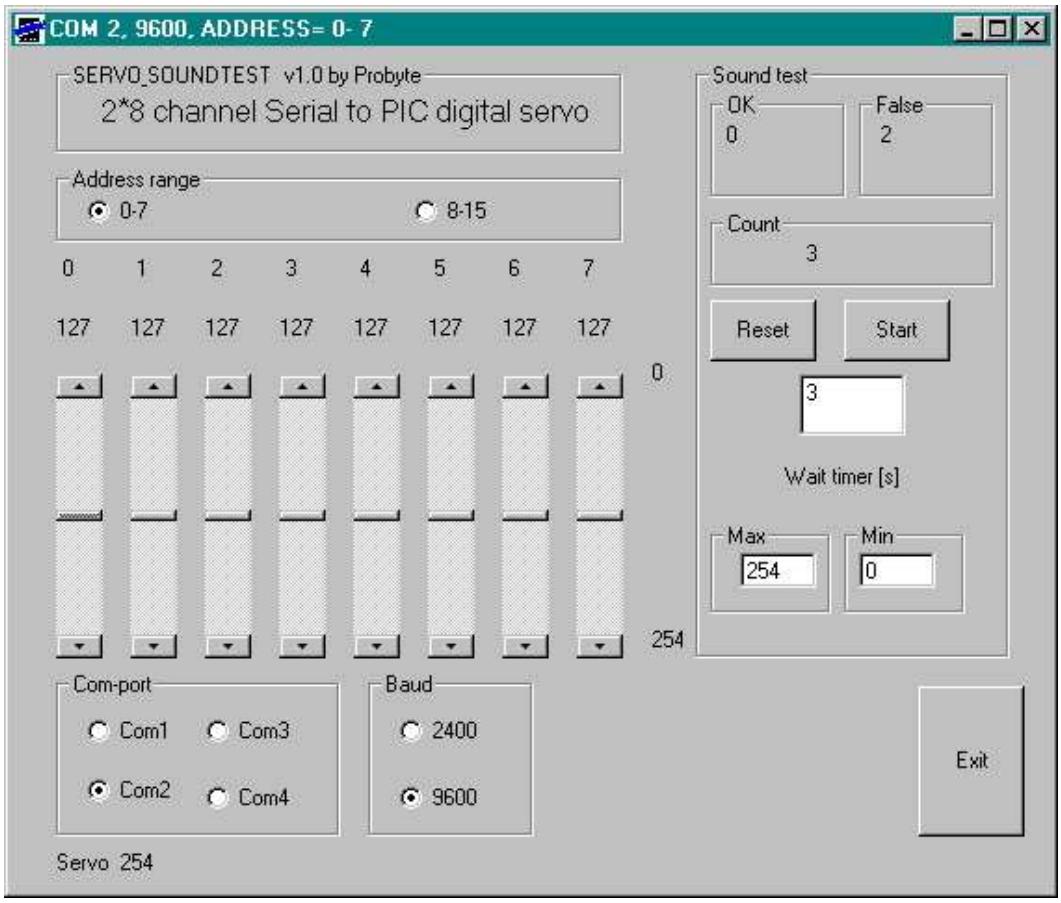


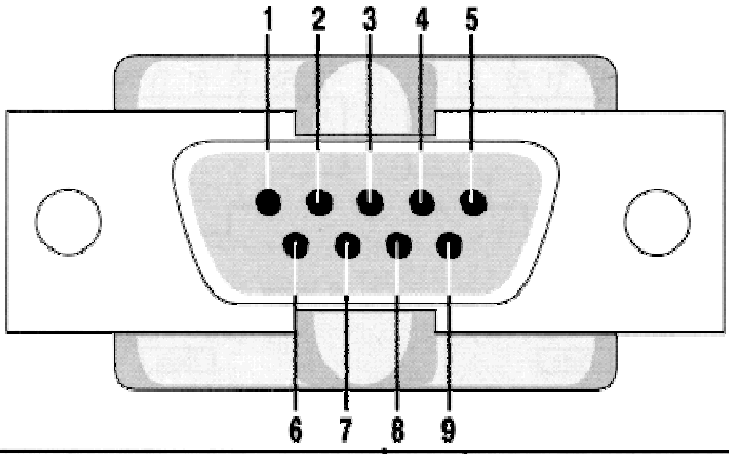
Illustration 3: PICSERVO software for Windows [18]

Analyzing how the N810 device should be connected with the control board, the N810 works with Linux instead of Windows, and moreover the communication port is USB, type micro USB. Thus arise the need to adapt the system from Windows/RS232 to Linux/USB.

Linux/RS232 configuration was implemented using a program to send information through the ports compatible with Linux. On the website of the PICSERVO control board are described the string of bytes that are required to send to control the servos. In this way the system was running properly.

Windows/USB configuration was implemented using a RS232/USB converter and the program that comes with the board. It did not work. It was

needed to know, at that point, what was sent through the pins of the serial port from the computer when using the converter RS232/USB, and compare the results with the serial port without using the converter. It employed an oscilloscope measuring pin to pin the signal being transmitted. The results of this analysis are reflected in table 1.



Pin	Signal	Pin	Signal
1	Data Carrier Detect	6	Data Set Ready
2	Received Data	7	Request to Send
3	Transmitted Data	8	Clear to Send
4	Data Terminal Ready	9	Ring Indicator
5	Signal Ground		

Illustration 4: RS232 pinout [19]

Name	Vpp	Vmax	Vavg
PIN 1 COM 1	63,13 mV	36,25 mV	15,24 mV
PIN 1 COM 6	33,44 mV	14,69 mV	1,692 mV
PIN 2 COM 1	56,25 mV	34,38 mV	15,77 mV
PIN 2 COM 6	35,94 mV	17,19 mV	1,357 mV
PIN 3 COM 1 (Tx)	2,438 V	1,187 V	-1,154 V
PIN 3 COM 6 (Tx)	1,375 V	718,7 mV	-594,8 mV
PIN 4 COM 1	52,50 mV	-1,130 V	-1,145 V
PIN 4 COM 6	92,19 mV	-550,0 mV	-592,4 mV
PIN 5 COM 1	54,38 mV	16,25 mV	0,28 mV
PIN 5 COM 6	60,62 mV	18,12 mV	0,78 mV
PIN 6 COM 1	48,13 mV	30,00 mV	15,59 mV
PIN 6 COM 6	31,88 mV	11,25 mV	1,035 mV
PIN 7 COM 1	48,13 mV	31,25 mV	15,74 mV
PIN 7 COM 6	100,00 mV	-553,1 mV	-592,0 mV
PIN 8 COM 1	56,25 mV	32,50 mV	15,77 mV
PIN 8 COM 6	36,56 mV	14,38 mV	1,075 mV
PIN 9 COM 1	48,13 mV	32,50 mV	15,81 mV
PIN 9 COM 6	35,63 mV	14,69 mV	1,002 mV

Table 1: Signal through the pins of the serial port

Vpp is the peak-to-peak voltage of the signal, Vmax is the maximum value and Vavg is the average value of the signal.

By measuring the pin number 3, Tx transmission, the values are different for the two cases, with the RS232/USB converter and without it. The value that is read from the Tx pin when it sends a pulse of information using the converter is significantly smaller, therefore the PICSERVO board does not detect it as a logical 1, and the servos are not moving.

It is after this analysis when deciding to use the AVR, the red board, which has many more possibilities and flexibility of programming. In making this choice arise the need to create the software for programming the AVR microcontroller to make it as efficient as possible to move the servos.

AVR MICROCONTROLLER BOARD

Antti Karjalainen explains the description and the operation of the AT90USB162-board.

> "AT90USB162-board has been specially designed for Space Master course (2008) at HUT. The board is cheap (costs under 20 €/each) and can be programmed directly from USB-port (without external programmer, such as STK500). The AT90USB162-chip has also internal regulator for 3.3 V. Normally the device uses the USB-port also as a power supply. Pins V+/GND can also be used as a power input (Illustration 5).
>
> The boards consist only of the essential components of AT90USB162 device. All free pins are connected to pinheads (Illustration 5 and Table 2). The voltage can be selected to 5 V (directly from USB) or 3.3 V from the internal regulator. The 16 MHz external crystal is needed for the USB-communication (boot loader) to work as expected. The boot loader is already programmed to the chip. The standard ISP (found for example from STK500) can also be used if necessary." [17]

The section about how to program the AVR is not the subject of this work, and is widely described in the paper called AT90USB162-board, written by Antti Karjalainen in HUT, in Espoo, Finland.

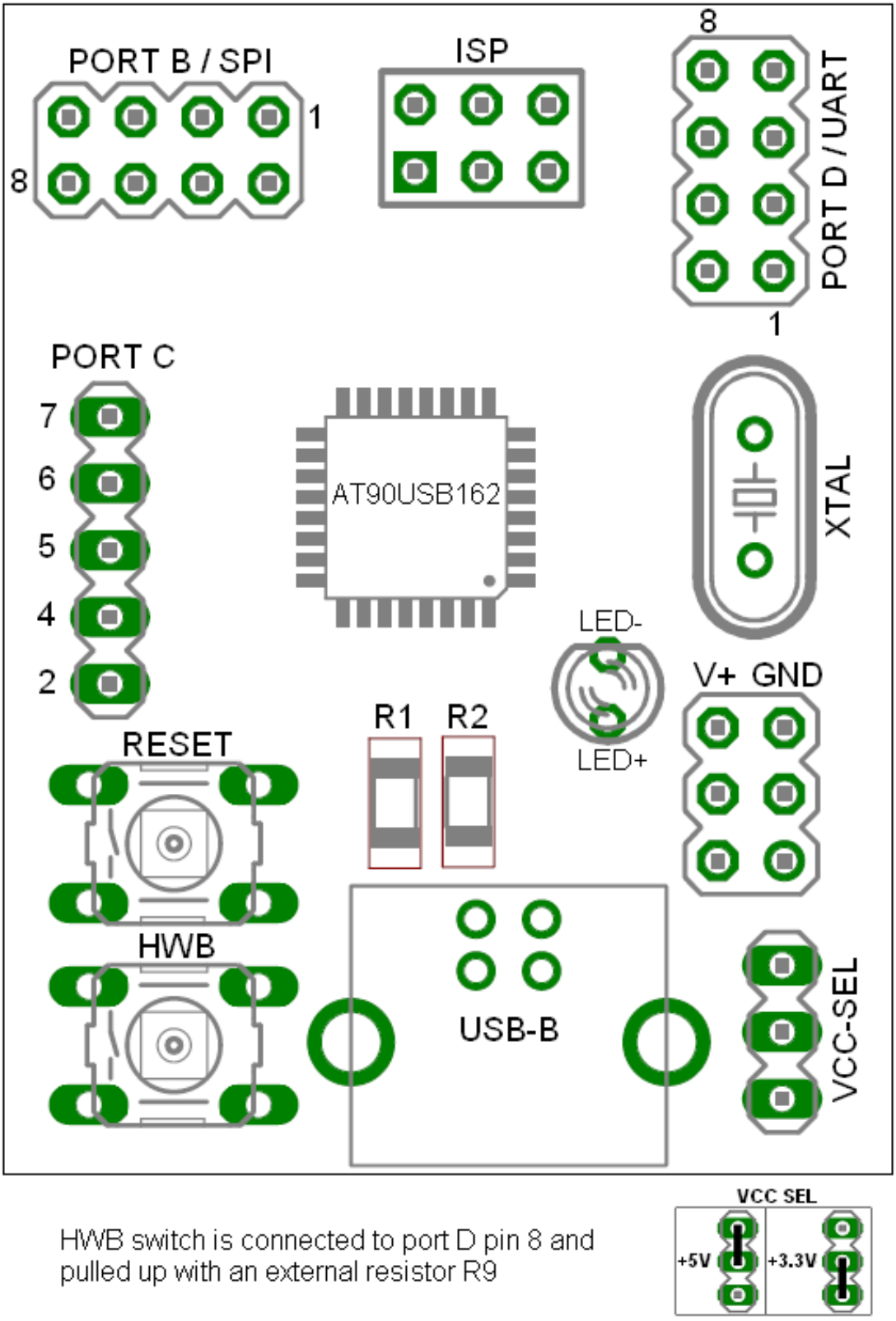


Illustration 5: AT90USB162-board [17]

Name	Description	Function
PB / SPI	Port PB 1–8	AT90USB162 datasheet - pin configurations
PC	Port PC 2, 4 ,5 ,6 , 7	AT90USB162 datasheet - pin configurations
PD / UART	Ports PD 1–8	AT90USB162 datasheet - pin configurations
ISP	ISP-connector	For programming
USB-B	USB-connector	Connector to USB-master device / Program port
V+ GND	Voltage selector	From USB-direct or 3.3 V internal regulator
VCC LED	LED-power on	The voltage indicator
VCC SEL	Voltage selector jumper	3.3 V / 5 V

Table 2: AT90USB162-board [17]

With the AVR we can control up to three servo motors simultaneously through the generation of PWM wave, where the different channels are:

Port	Channel
C/6	1
C/5	2
B/8	3

Table 3: Correspondence between ports and channel

THE SOFTWARE

To program the AVR microcontroller is been used a code developed by the company of microcontrollers ATMEL. The software has been used before many times and trouble-free operation is verified. The initial problems have been resolved with time and with the aid of experience.

In the next chapter it follows an explanation of how works the software used to program the microcontroller of the AVR.

> "The application entry point is located is the main.c file. The main function first performs the initialization of a scheduler module and then runs it in an infinite loop. The scheduler is a simple infinite loop calling all its tasks defined in the conf_scheduler.h file. No real time schedule is performed, when a task ends, the scheduler calls the next task defined in the configuration file (conf_scheduler.h).
>
> The sample dual role application is based on two different tasks:

- The usb_task (usb_task.c associated source file), is the task performing the USB low level enumeration process in device mode. Once this task has detected that the usb connection is fully operationnal, it updates different status flags that can be check within the high level application tasks.

- The hid task performs the high level device application operation. Once the device is fully enumerated (DEVICE SET_CONFIGURATION request received), the task checks for received data on its OUT endpoint and transmit data on its IN endpoint." [22]

This software is independent of the operating system of the computer or device used to work with the AVR, and it is very robust software and an ideal solution for the problem that is dealing here.

To move the servo to the desired position a PWM signal is used, and is generated by the AVR's microcontroller. By varying the pulse width of the PWM signal, that is, by varying the duty cycle, it reached the different positions the servos are able to into a certain range. The range covered by the servo is approximately 200 degrees. This range will be sufficient if the robot is placed near a wall; if the robot is not placed in the middle of the room.

Matti Öhman explained the generation of the PWM and SWP signals to control the servos as follows.

"To control the generated PWM the following commands must be used:

- $PWM,1000,1500,2000*
- $SWP,1000,1500,2000*

The first command will set the PWM pulse lengths to 1000us, 1500us and 2000us for channels 1, 2 and 3 respectively. The default pulse width is 1500 us at the power up. Specifying values greater than 20000us will result undefined behaviour (PWM > 100%) and is not recommended.

The difference between the PWM and SWP commands is that the first sets the indicated values instantly while the second uses slow ramps (or sweeps) to transition to the desired pulse lengths. The change rate for pulse lengths is 50us/s, so it will take 20 seconds to get from 1000us to 2000us.

It is also possible to use the shorthand versions of the commands:

- $PWM,1000,,* // instantly sets the first channel to 1000us
- $SWP,,,2000* // gradually sets the third channel to 2000us ”

The pulse widths are checked so it is possible to specify values which are inside the recommended range (400us – 2350us).

2.3 Communication between N810 and AVR

The N810 device is the “brain” of the robot. Is the element responsible for interacting with the outside world and transform the orders given in commands capable to move the servo motors of the robot. The communication between the N810 and the AVR is essential for the robot to reach its goal in a satisfactory way.

Thanks to the selection of the AVR board instead of the PICSERVO board, working directly using the USB ports of the computer is possible, without using any RS232/USB converter. Thus, at the time to work finally with the N810 device the change in the code will be minimal, since it will be implemented and verified the functionality of the Linux/USB configuration.

THE SOFTWARE

The program’s code to control the servo motors has been developed by Gonzalo Zubieta in HUT, and has been tested successfully in a Linux PC. The first tests to implement the software on the N810 are underway at the same time that this document is being written. Therefore, the results nowadays are not yet satisfactory and the final code can be modified from the original code to correct errors not detected during the operation through the PC.

The code is composed of the following functions:

- Opening the USB port of the device for sending and receiving signals through it.
- Writing the commands to control the servos as required by movement.
- Creation of a loop to continue writing commands, selecting the channels and controlling the servos.
- Closing the USB port and termination of communication.

```
#include <stdio.h>   /* Standard input/output definitions */
#include <string.h>  /* String function definitions */
#include <unistd.h>  /* UNIX standard function definitions */
#include <fcntl.h>   /* File control definitions */
#include <errno.h>   /* Error number definitions */
#include <termios.h> /* POSIX terminal control definitions */

main(void)
   {
      int fd; /* File descriptor for the port */
      int n,pl;
      char c;
      char type [5];
      char buf[30];
      size_t nbytes;


      fd = open("/dev/ttyACM0", O_RDWR | O_NOCTTY | O_NDELAY);
/* The name of the port can be different from ttyACM0 */

      if (fd == -1)
      {
     /* Could not open the port */

      perror("open_port: Unable to open /dev/ttyACM0 - ");
      }

      else
      {
      fcntl(fd, F_SETFL, 0);
      printf("Port open\n");




 /* Loop for writing in the port */
      while(c!='0')
            {
            printf("Insert the channel to control (1,2,3). For exit
                    insert 0: ");
            scanf("%s",&c);

            switch(c)
                  {
                  case '1':
                        printf("Insert the type of the signal (PWM or
                                SWP): ");
                        scanf("%s",type);
                        printf("Insert the pulse length in us
                               (400-2350): ");
                        scanf("%d",&pl);

                        sprintf(buf, "$%s,%d,,*",type,pl);
                        nbytes = strlen(buf);

                        printf("buf = %s\n",buf);
                        /* Shows what's on buf */

                        n=write(fd, buf, nbytes);
/* Writes the content of buf in the port with fd as file descriptor */
```

```
                        if (n < 0)
                              fputs("write() of 4 bytes
                                    failed!\n", stderr);
                              break;


                  case '2':
                        printf("Insert the type of the signal (PWM or
                               SWP): ");
                        scanf("%s",type);
                        printf("Insert the pulse length in us
                               (400-2350): ");
                        scanf("%d",&pl);

                        sprintf(buf, "$%s,,%d,*",type,pl);
                        nbytes = strlen(buf);

                        printf("buf = %s\n",buf);
                        n=write(fd, buf, nbytes);

                        if (n < 0)
                              fputs("write() of 4 bytes
                                    failed!\n", stderr);
                              break;

                  case '3':
                        printf("Insert the type of the signal (PWM or
                               SWP): ");
                        scanf("%s",type);
                        printf("Insert the pulse length in us
                               (400-2350): ");
                        scanf("%d",&pl);

                        sprintf(buf, "$%s,,,%d*",type,pl);
                        nbytes = strlen(buf);

                        printf("buf = %s\n",buf);
                        n=write(fd, buf, nbytes);

                        if (n < 0)
                              fputs("write() of 4 bytes
                                    failed!\n", stderr);
                              break;

                        default:
                              printf("Channel not found\n");
                              break;
                  }
            }
      }

      return (fd);

      close(fd);
    }

/* End of the code */
```

3. Results

3.1 Reached solutions

The main results of this project are two:

- The control of the servo motors using the AVR Microcontroller board.
- The development of an application to handle the servo motors from a device running Linux.

Due to the work done in this project it has achieved to develop and test a solution to control the servos, responsible for move the N810. On the choice of AVR Microcontroller board based on the experience of the department of Automation and Systems Technology, it has developed a software to program the microcontroller on the board and achieve the goal initially proposed. Through some basic commands the AVR is able to generate the necessary waves to control the servos and move them to the desired position.

On the other hand, it has developed an application to make easy the control of the servos from a device running Linux. The servos's control from a PC is a success. It also needed to create a communication between the N810, which works with Linux, and the AVR.

It's here where the main problem appears when trying to establish communication. The N810 is a device, though with a simple software can become a host of communication. When using the N810, and not a PC to send signals, the AVR must be externally powered by a power supplier. The problem reside in the cable used for physical communication. The cable is kind of "peripheral" rather than "host". That is, attempting to establish communication the energy flows from the AVR to the N810, and not on the opposite way, as it should occur; thus N810 crashes and restarts again and again. The solution to this problem is to purchase the correct cable or modify the current so it can work in the necessary circumstances.

3.2 Future work

The vision of Ilonani, a tablet-top robot for remote interaction, might want to be revised. Nevertheless, if it's considered to be current when new work is to

be undertaken, it might be useful to follow the work done on this project, since many false trails have been uncovered and much work could be reused.

Furthermore, the code and experience summarized in this document could be reused in the coding of a final application, in order to save time and effort.

SYSTEM COMPONENTS

In the chapter of describing the system components it has been discussed about several blocks that are part of the robot but have not been developed in this project. On one side it has commented briefly to have all the adaptation and power concentrated in one electronics board. The development of this power board needs further investigation to design and optimize its performance, maximizing yield and minimizing the loss of power on the robot.

On the other hand it has also discussed the need to use a battery to supply the robot for a limited time, thus becoming a wireless element. A preliminary analysis concluded the approximate values of the capacity and power for the system, but this analysis should be reviewed since it has not yet been worked with the complete system neither with the real demand for power.

Finally, it is thought the implementation of a few buttons, either through hardware or software, to control the movements of the robot manually. This block needs to be fully developed, building the buttons if they are kind of hardware, or developing a new code when using software buttons. Both cases will require the implementation of a code to manage orders and communicate them to the servo motors via the control board.

BLUETOOTH. THE ALTERNATIVE COMMUNICATION

To transfer data to and from the Nokia N810, Bluetooth is also available. This project explores the communication by USB, but it is worth noting that this is not the only communication possible between the N810 and the AVR.

The Maemo project explained the Bluetooth communication as follows.

> "The N810 device has a built-in Bluetooth radio that allows doing things like tether to a cellphone for a data connection, transferring files between devices, and connecting input devices.
>
> There are two Bluetooth networking protocols. The first, DUN, is well-supported in Maemo and is the recommended method for tethering to a cellular device. The second, PAN, does not have official support, but is well-supported by the community. PAN is faster and more versatile, but supported by fewer devices than

> DUN. DUN is the primary method for tethering to cellular devices. It is easy to set up and easy to use, but slower than PAN in many cases. PAN is faster and more versatile than DUN, though supported by fewer devices, and not officially supported in Maemo, but support is provided by the community package. PAN is particularly useful for setting up Bluetooth networks between your computer and your tablet." [21]

To use this communication is needed to provide a Bluetooth communication system to the AVR. The current version of the AVR lacks this system, but is proposed as a possible future work, or a more evolved version of the communication objective of this project, develop the code that allows the successful routing of commands between the N810 and AVR.

References

[1] Horizon Report 2009,
http://www.nmc.org/pdf/2009-Horizon-Report-K12.pdf
(checked June 15, 2009)

[2] http://www.daylife.com/photo/0b1OfUf2ZaaTM
(checked June 15, 2009)

[3] Ambient devices philosophy,
http://www.ambientdevices.com/cat/philosophy.html
(checked June 15, 2009)

[4] http://www.telemedicineinsider.com/
(checked June 15, 2009)

[5] http://www.aethra.com
(checked June 15, 2009)

[6] “The shape of things to come”, Population Action International,
http://www.populationaction.org/Publications/Reports/The_Shape_of_Things_to_Come/Summary.shtml
(checked June 15, 2009)

[7] http://www.cienladrillos.com/2008/02/13-una-interfaz-de-voz-al-servicio-de-los-discapacitados
(checked June 15, 2009)

[8] http://www.easy-life.es/
(checked June 15, 2009)

[9] http://ambientdevices.com/cat/index.html
(checked June 15, 2009)

[10] http://www.avcorporate.com/videoconf.htm
(checked June 15, 2009)

[11] http://www.sfgate.com/cgi-bin/article.cgi?f=/c/a/2003/10/06/BUGUO24GQC1.DTL
(checked June 15, 2009)

[12] http://en.wikipedia.org/wiki/Internet_Tablet
(checked June 15, 2009)

[13] http://en.wikipedia.org/wiki/Nokia_N810
(checked June 15, 2009)

[14] http://europe.nokia.com/find-products/devices/nokian810/specifications
(checked June 15, 2009)

[15] http://www.crunchgear.com/2008/04/01/nokia-officially-unveils-the-n810-wimax-edition/
(checked June 15, 2009)

[16] http://en.wikipedia.org/wiki/Maemo_(operating_system)
(checked June 15, 2009)

[17] AT90USB162 evaluation board. User Guide v.0.1 draft, Antti Karjalainen (2008)

[18] http://www.probyte.fi/catalog/product_info.php?products_id=1103&osCsid=1054250af1adf5
(checked June 15, 2009)

[19] http://www.aggsoft.com/rs232-pinout-cable/serial-cable-connections.htm
(checked June 15, 2009)

[20] http://www.easysw.com/~mike/serial/serial.html#2_5
(checked June 15, 2009)

[21] http://wiki.maemo.org/Bluetooth
(checked June 15, 2009)

[22] http://www.atmel.com/dyn/products/tools_card.asp?tool_id=3886
(checked June 15, 2009)

Appendix A. Ilonani concept

This Chapter deals with the Ilonani concept and the system components, and it can be found in Raúl Rodriguez Pearson's final project (2009). Raúl Rodriguez was working closely to Gonzalo Zubieta on a big project called Ilonani, where this work is a part of it.

A.1 The concept and goals

Having the previous section in mind, the goal for the project started by designing and constructing a table-top robot providing a friendly user interface for elderly people to aid in the supply of tele-assistance services. The list of goals pursued has been:

- Research the possibility of using Nokia's N810 internet tablet as the platform for the robots "brain" and "face". Finding out if videoconferencing is feasible (taking into account different application scenarios) and if it provides potential for a friendly enough user interface (given our requirement that it should be usable for elderly people).

- If the Nokia is sufficient, research what possible solutions exist, what different paths can be followed to create a working system and implement the best solution.

- Design and build a physical prototype of the robot, including a charging base, a case hosting all the necessary electronics and a turn-and-tilt mechanism.

The original vision also included the functionality of tracking the speaker with the camera but after some research, this was deemed too difficult. Additionally, the original project was split in two, so this project will actually take care of the third goal, concerning the issues around the design and building of a physical prototype. More information about the video-conferencing solution and the N810 will be found in Raúl Rodríguez's Final Project, which has been written in HUT, in the spring of 2009.

EXTENSIBILITY AND BASIS FOR FUTURE WORK

Another objective that has been kept in mind is trying to develop a design that will be extensible. Since the beginning it has been obvious that a lot of great ideas (of development and application for Ilonani) would have to be set aside since it would not be possible to implement all of them in the time given. Pursuing this goal means that a lot of extra research has been done in order to set the basis for future projects wanting to build on this one. Hopefully, it also means that the information written in this document will be easy to follow and put into practice.

A.2 System components

The system will consist of the components depicted in Illustration 6. As already stated, Nokia's N810 will be the main “brain” and “face” of the robot. It will have USB connection with an AVR microcontroller card used to generate the low level PWM signals to control the turn-and-tilt servos. Additionally, we will use a power card to concentrate the electronics needed to adapt and provide the supply lines to power the AVR microcontroller card and servos. The power card will also provide an interface with the battery (to recharge it or to draw power from it, depending if the robot is connected or not to a power line through the power connector). We also decided to power the Nokia tablet using its own power connector, to make the system simpler and prevent any damage from occurring.

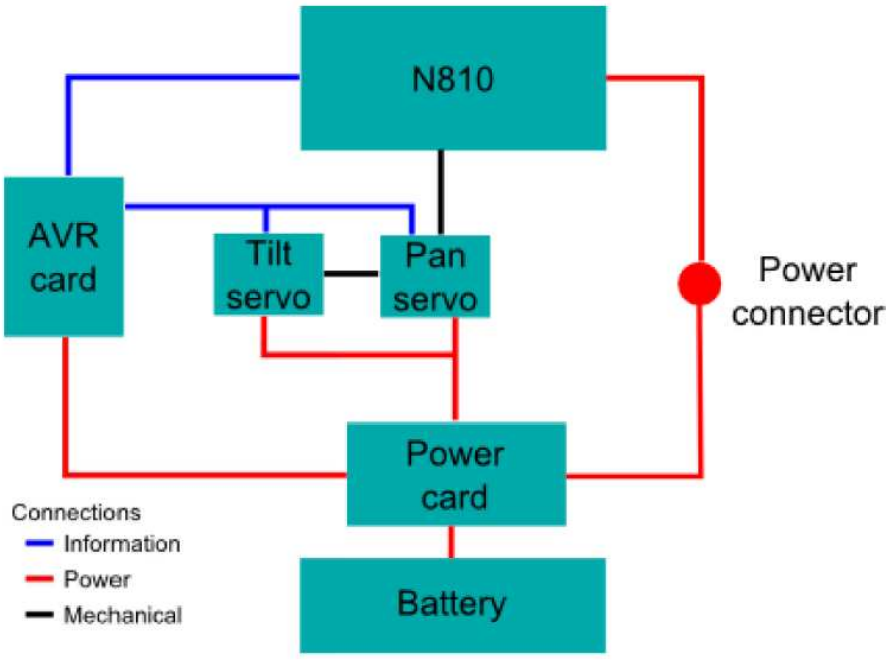


Illustration 6: Illonani block diagram

NOKIA'S INTERNET TABLET, THE N810

Nokia's N810 is an internet tablet, a kind of internet appliance (a device aimed mainly at accessing internet services) developed by Nokia under the N series. These internet tablets should not be considered as mobile phones, since they have no way of connecting to any cellular network typical for mobile phones. Nevertheless, they provide Wi-Fi and Bluetooth connectivity that enable access to the internet (with the latter, when used in conjunction with a web-enabled mobile phone) [12].

A list of the N810's technical specifications can be found in Table 4.

Size ● Volume: 128 cc ● Weight: 226 g ● Length: 72 mm ● Width: 128 mm ● Thickness: 14 mm
Display ● High-resolution 4.13" WVGA display (800 x 480 pixels) with up to 65,000 colours
Processor ● TI OMAP 2420, 400Mhz
Memory ● DDR RAM 128MB ● Flash 256MB
Storage ● Up to 2GB internal memory ● Support for compatible miniSD and microSD memory cards (with extender). Supports cards up to 8GB. (SD cards over 2GB must be SDHC compatible.)
Operating Times ● Battery: Nokia Battery BP-4L ● Continuous usage (display on, wireless LAN active): up to 4 hours ● Music playback: up to 10 hours ● Always online time: up to 5 days ● Standby time: up to 14 days
Other characteristics ● Smooth slide with integrated QWERTY keyboard ● Built-in GPS receiver ● High quality stereo speakers and sensitive microphone ● High-resolution wide-screen display ● Integrated desk stand ● Integrated VGA web camera ● HW key to lock touch screen and keys ● Ambient light sensor
Connectivity ● WLAN standard: IEEE 802.11b/g ● Bluetooth specification v.2.0 . +EDR (profiles supported: HID, FTP, DUN, GAP, SPP, HSP, SAP and OPP) ● USB high speed for PC connectivity ● 3.5 mm stereo headphone plug (Nokia AV Connector)

Table 4: N810's technical specifications [14]

Historically, it's the third version of Nokia's product, preceded by the 770 and the N800. The first version, the 770, was presented in May of 2005, and

received many critics, mainly due to its slowness and its short battery-life. This model evolved into the N800, introduced in January of 2007, which addressed many of the 770's shortcomings. That same year, in October, the N810 was released. A WiMAX enabled edition was announced on April of 2008, but its production was cancelled in January of 2009, probably due to the low availability of WiMAX providers [13].

The tablets run a modified version of Debian GNU/Linux called Maemo. It's a custom made operating system for Nokia Internet Tablets. Similarly to other tablet operating systems, it features a home screen with a menu bar, an application-launching area and a customizable “desktop” where different applets, ranging from RSS readers to clocks, can be run.

Illustration 7: Nokia N810 [15]

It's heavily based on the GNOME project, from which it draws many libraries, GUI components and application frameworks (like GStreamer, used for multimedia handling). It uses Matchbox as window manager and the GTK-based Hildon application framework for GUI elements [16].

Maemo offers a very wide range of software applications – from telecommunication applications (for Internet, instant messaging and VoIP), office and media applications to games and a huge range of applications common in Linux desktop PCs.

As shown in Illustration 6, the N810 is intended to be connected with an AVR microcontroller card (developed in the Automation Technology Laboratory) using a USB interface. The goal is to implement some code to control the movement of the turn-and-tilt mechanism from the tablet. This code should be

developed so that it will be embeddable in applications developed for the N810. For example, as part of the user interface, some virtual buttons could be presented on the screen to allow the user to pan or tilt the robot's head (which would be the Nokia itself). It could also allow the implementation of a tele-controlled system, where a remote user controls the movement of the system by sending commands to the software running on the tablet.

AVR MICROCONTROLLER BOARD

The AVR microcontroller board has been developed by Antti Karjalainen at the Department of Automation and Systems Technology of TKK. It features an AT90USB162 microcontroller from ATMEL.

Illustration 8: AVR microcontroller board [17]

Relevant to this discussion, it has already been successfully used within the Laboratory to control sets of servos from Linux machines. The only work left would be to adapt the systems and Linux drivers to the Maemo environment.

Additionally, using this board provides the potential to extend the system through the use of the Zigbee network developed under a different project within the Laboratory.

POWER SUPPLY BOARD

An idea considered has been to have all the power adaptation and electronics concentrated in one board. In an initial vision, this would include:

- Electronics that measure the charge of the battery: this gives the information to operate the low battery indicator and to operate the recharging of the battery (that should stop when a certain voltage level is reached, that is, the battery is properly charged).
- Low battery LED indicator.
- Drivers for servos.
- Power adaptation to provide supply for AVR microcontroller board.

- Connector where a 220V AC to 12V DC transformer could be plugged in, to provide connection to the mains. Connectors for battery, AVR microcontroller board and servos.

Not all features might be implemented. Some of them need more time to develop an initial prototype, and can be taken as future work.

SERVOS

The servos used to build the turn-and-tilt mechanism are the classical Futaba S3003. They provide an affordable and reliable solution. Using a PWM signal, they can be position controlled.

Illustration 9: S3003 servo

Dimension	40.4 mm x 19.8 mm x 36.0 mm
Weight	37.2 g
Operating Speed	0.23 sec/60° (4.8V) 0.19 sec/60° (6.0V)
Output torque	3.2 kg-cm (4.8V) 4.1 kg-cm (6.0V)

Table 5: Futaba S3003 specifications

BATTERY

From a preliminary study, the peak current demanded by each servo (when blocked) is around 1,3 Amperes. On the other hand, nominal consumption of the whole system was found to be around 1 Ampere. The initial specification required batteries to support 6 hours of normal operation before needing a recharge. The battery, as the power supply board, is not developed in this project, and creates another path to follow in a future work.

BUTTONS

The idea of adding physical buttons was also pondered. On/off and movement buttons seem to be a minimum. They would add usability, but they would also make the system more complex.

Hardware buttons:

- Advantages: they are fixed and don't move while pressing.
- Disadvantages: more complicated to build, difficult to add multifunctionality, not so flexible.

Software buttons:

- Advantages: they are more flexible, the information is easily captured, processed and sent to AVR microcontroller as commands. All of the “thinking” happens inside the tablet.
- Disadvantages: they move while being pressed (the tablet moves, so it's not so comfortable).

Another solution would be to implement the buttons via software. Each solution has its own advantages and disadvantages.

Appendix B. Accessing serial ports

In this chapter is explained a short guide to program the serial ports of a UNIX workstation. The full guide about this matter can be found in The Serial Programming Guide for POSIX Operating Systems [20].

Like all devices, UNIX provides access to serial ports via *device files*. To access a serial port you simply have to open the corresponding device file.

SERIAL PORT FILES

Each serial port on a UNIX system has one or more device files (files in the /dev directory) associated with it:

System	Port 1	Port 2
IRIX®	/dev/ttyf1	/dev/ttyf2
HP-UX	/dev/tty1p0	/dev/tty2p0
Solaris®/SunOS®	/dev/ttya	/dev/ttyb
Linux®	/dev/ttyS0	/dev/ttyS1
Digital UNIX®	/dev/tty01	/dev/tty02

Table 6: Serial Port Device Files

OPENING A SERIAL PORT

Since a serial port is a file, the *open()* function is used to access it. The one hitch with UNIX is that device files are usually not accessible by normal users. Workarounds include changing the access permissions to the file(s) in question, running your program as the super-user (root), or making your program set-userid so that it runs as the owner of the device file (not recommended for obvious security reasons...)

For now we'll assume that the file is accessible by all users. The code to open serial port 1 on a PC running Linux is show next:

Listing 1 - Opening a serial port.

```
#include <stdio.h>   /* Standard input/output definitions */
#include <string.h>  /* String function definitions */
#include <unistd.h>  /* UNIX standard function definitions */
```

```
#include <fcntl.h>   /* File control definitions */
#include <errno.h>   /* Error number definitions */
#include <termios.h> /* POSIX terminal control definitions */


/*
 * 'open_port()' - Open serial port 1.
 *
 * Returns the file descriptor on success or -1 on error.
 */

int
open_port(void)
{
  int fd; /* File descriptor for the port */


  fd = open("/dev/ttyS0", O_RDWR | O_NOCTTY | O_NDELAY);
  if (fd == -1)
  {
   /*
    * Could not open the port.
    */

    perror("open_port: Unable to open /dev/ttyS0 - ");
  }
  else
    fcntl(fd, F_SETFL, 0);

  return (fd);
}
```

Other systems would require the corresponding device file name, but otherwise the code is the same.

When the device file is opened it uses two other flags along with the read+write mode:

```
fd = open("/dev/ttyS0", O_RDWR | O_NOCTTY | O_NDELAY);
```

The *O_NOCTTY* flag tells UNIX that this program doesn't want to be the "controlling terminal" for that port. If this isn't specified then any input (such as keyboard abort signals and so forth) will affect the process. Programs like *getty(1M/8)* use this feature when starting the login process, but normally a user program does not want this behavior.

The *O_NDELAY* flag tells UNIX that this program doesn't care what state the DCD signal line is in, whether the other end of the port is up and running. If you do not specify this flag, your process will be put to sleep until the DCD signal line is the space voltage.

WRITING DATA TO THE PORT

Writing data to the port is easy, just use the *write()* system call to send data it:

```
n = write(fd, "ATZ\r", 4);
if (n < 0)
  fputs("write() of 4 bytes failed!\n", stderr);
```

The *write* function returns the number of bytes sent or -1 if an error occurred. Usually the only error that will run into is *EIO* when a MODEM or data link drops the DCD line. This condition will persist until the port is closed.

READING DATA FROM THE PORT

Reading data from a port is a little trickier. When you operate the port in raw data mode, each *read()* system call will return the number of characters that are actually available in the serial input buffers. If no characters are available, the call will block (wait) until characters come in, an interval timer expires, or an error occurs. The *read* function can be made to return immediately by doing the following:

```
fcntl(fd, F_SETFL, FNDELAY);
```

The *FNDELAY* option causes the *read* function to return 0 if no characters are available on the port. To restore normal (blocking) behavior, call *fcntl()* without the *FNDELAY* option:

```
fcntl(fd, F_SETFL, 0);
```

This is also used after opening a serial port with the O_NDELAY option.

CLOSING A SERIAL PORT

To close the serial port, just use the *close* system call:

```
close(fd);
```

Closing a serial port will also usually set the DTR signal low which causes most MODEMs to hang up.